Almacenamiento en sistemas computacionales

Universidad Tecnológica Nacional – FRBA

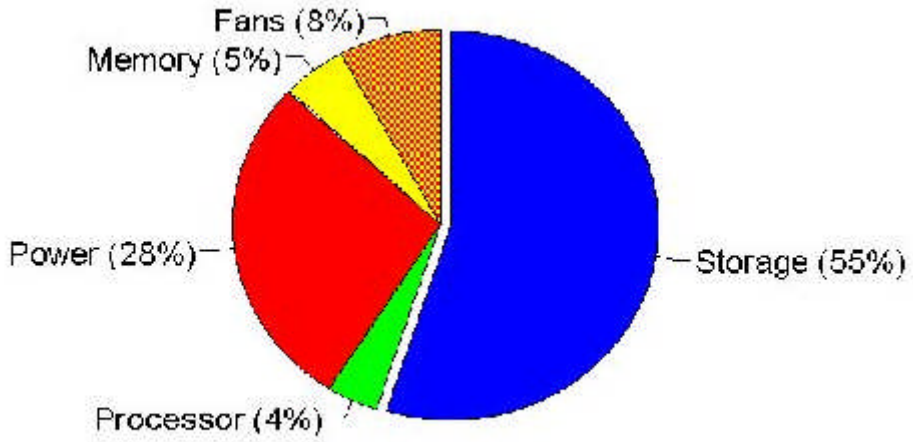
Autor: Gustavo Nudelman

Que es un sistema de almacenamiento?

- Es una organización de recursos de software y hardware para almacenar datos.
- Dichos recursos forman una entidad virtual que que se presenta al sistema operativo, aplicaciones y usuarios
- Es responsabilidad del ingeniero organizar estos recursos para brindar la solución mas eficiente al negocio como tambien cumplir con el SLA acordado.

Evolución de las Tecnologías de Almacenamiento

- En la década del 90 los sistemas de almacenamiento experimentaron una gran demanda en capacidad y disponibilidad (Internet, e-comerce, etc, transacciones 24x7)
- Necesidad de capacidad bajo demanda
- La tecnología de almacenamiento es la principal causa de caída de un servicio (Obsérvese el siguiente estudio)



SCSI

- Tiene Origen a principios de los años 80 cuando un fabricante de discos desarrollo su propia interfase de E/S denominado SASI (Shugart Asociates System Interface) que debido a su gran éxito comercial fue presentado y aprobado por ANSI en 1986.

- Podríamos definir SCSI como un subsistema de E/S inteligente y bidireccional

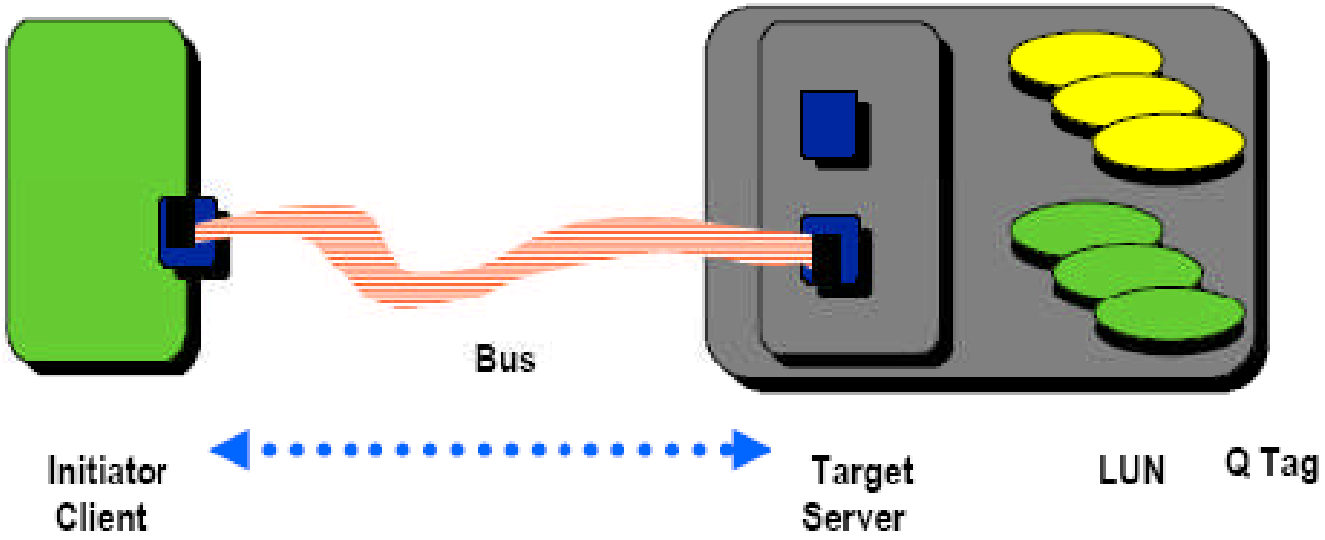
- Cada dispositivo del bus se direcciona en forma lógica con un ID y posee su propia Extensión de BIOS.

- Es una estructura de BUS – Sistema de contención – Un solo dispositivo toma el bus

- Según las diferentes versiones de SCSI se puede direccionar una cantidad n de dispositivos

- **SCSI Además de ser una especificación física de un subsistema de bus paralelo, es un potente protocolo que hoy en día trasciende el estar supeditado a una determinada arquitectura**

SCSI - Topología

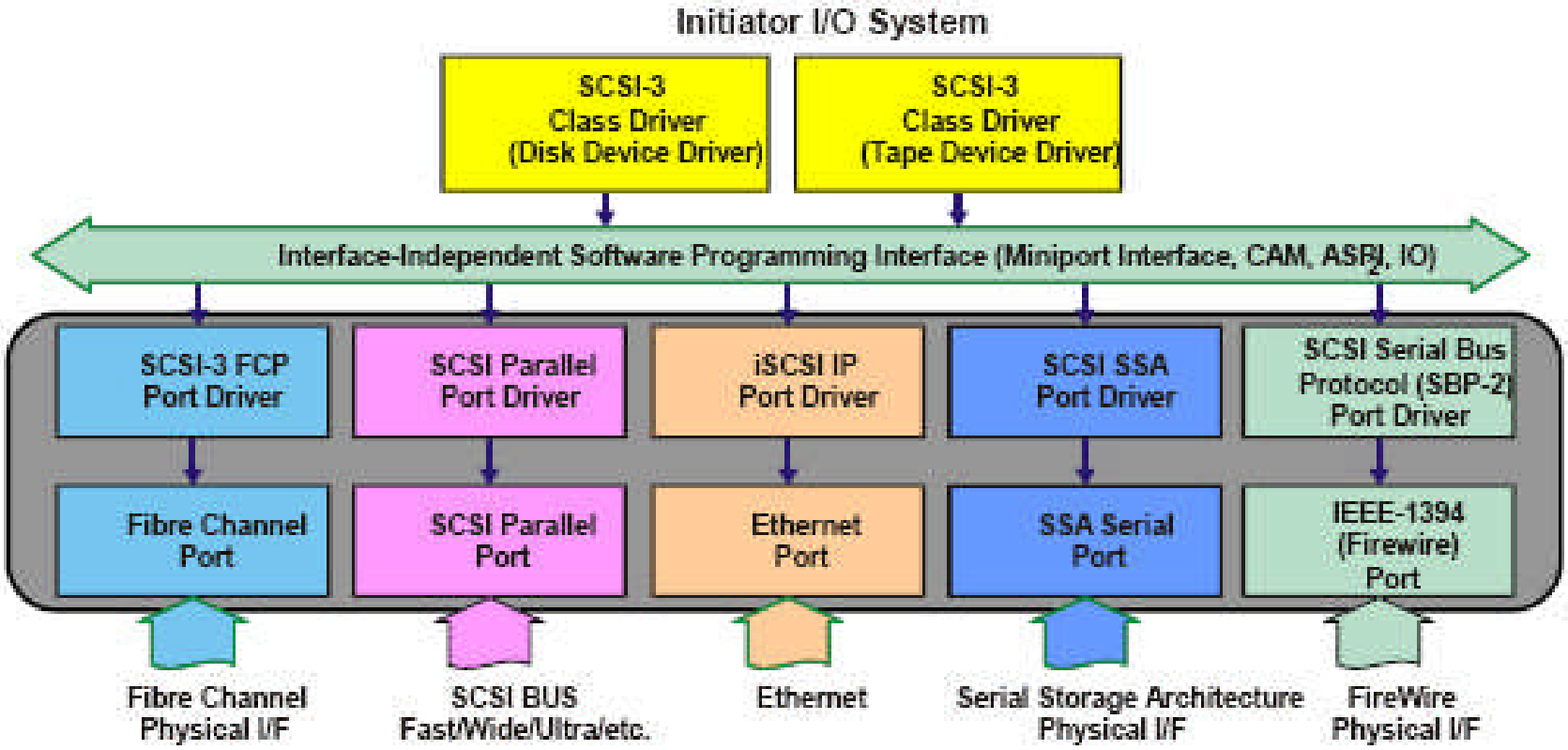


•Los dispositivos identificados pueden funcionar como Iniciadores o receptores

•Mediante un ID de dispositivo y un LUN se identifica al dispositivo en si mas una unidad lógica dentro del mismo.

•Cualquier dispositivo puede funcionar como Initiator y existir comunicación etre dispositivos del bus

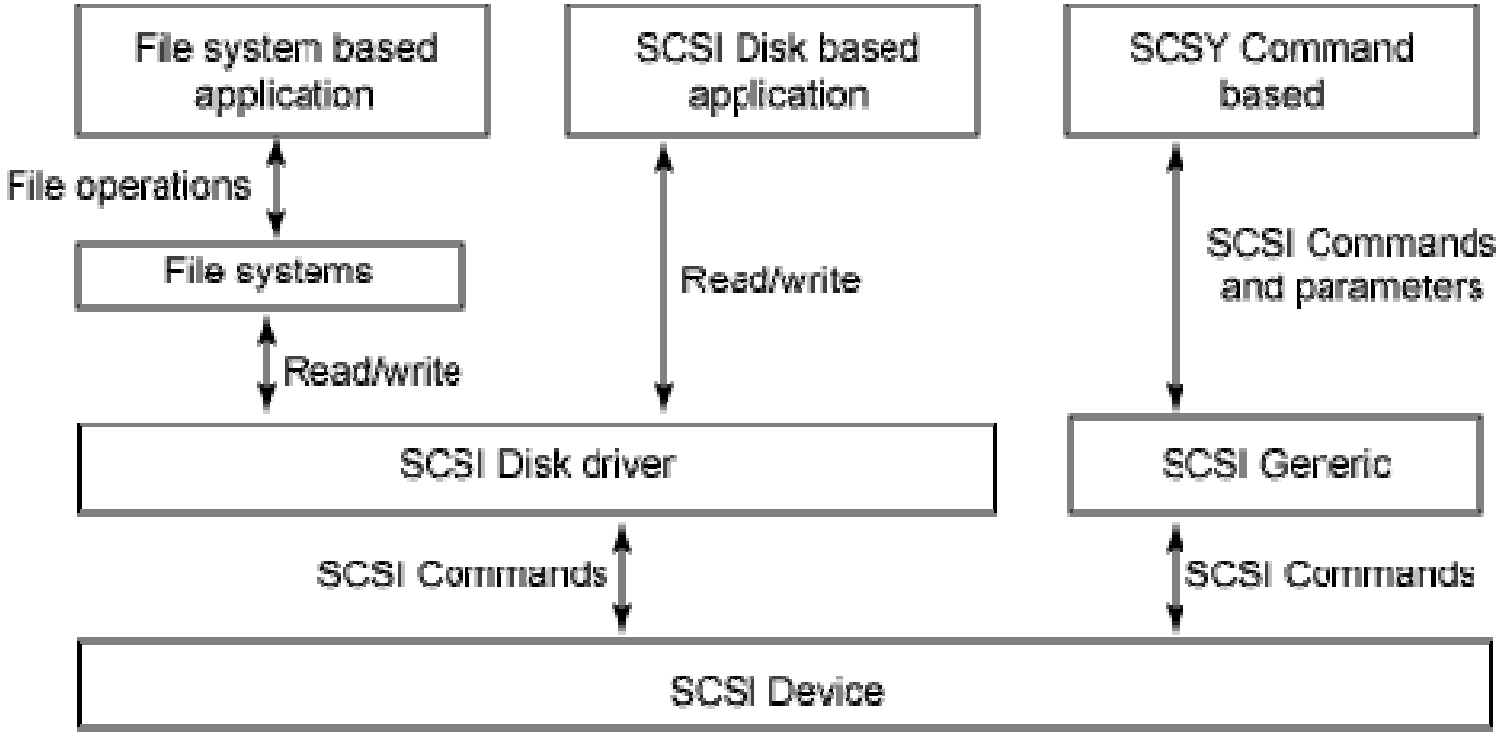
El protocolo SCSI es independiente del medio



SCSI – Operaciones I/O

- Si bien los comandos SCSI fueron definidos para el BUS paralelo creado con el standard, ha sido adaptado con mínimos cambios conexiones Serie punto a punto como Fibre Channel
- El nodo “Iniciador” envía un comando al nodo “Receptor” compuesto por una trama llamada “Command descriptor block” cuyo primer byte es el código de operación y 5 o mas bytes de argumentos dependiendo del comando
- El comando va dirigido a una determinada entidad LUN que es la “única” a la que se dirigen la operaciones I/O
- Según el comando se enviaran tramas de datos en cualquiera de ambas direcciones
- El Receptor enviará tramas de status luego de cada transacción
- **El driver SCSI construye un CDB (Command descriptor block) con las peticiones realizadas por la aplicación y los envía a la capa de transporte iSCSI. El driver SCSI también recibe CDBs de la capa iSCSI y envía los datos a la capa de aplicación.**

Modos de acceso a un dispositivo SCSI



Tipos de Comandos SCSI

Command	Description
Inquiry	Requests general information of the target device
Test/Unit/Ready	Checks whether the target device is ready for the transfer operation
READ	Transfers data from the SCSI target device
WRITE	Transfers data to the SCSI target device
Request Sense	Requests the sense data of the last command
Read Capacity	Requests the storage capacity information

Enviado por el Initiator en caso de que el status no sea OK

Ejemplo de operación “read”

bit→ ↓byte	7	6	5	4	3	2	1	0
0	Operation code = 08h							
1	LUN			LBA				
2	LBA							
3	LBA							
4	Transfer length							
5	Control							

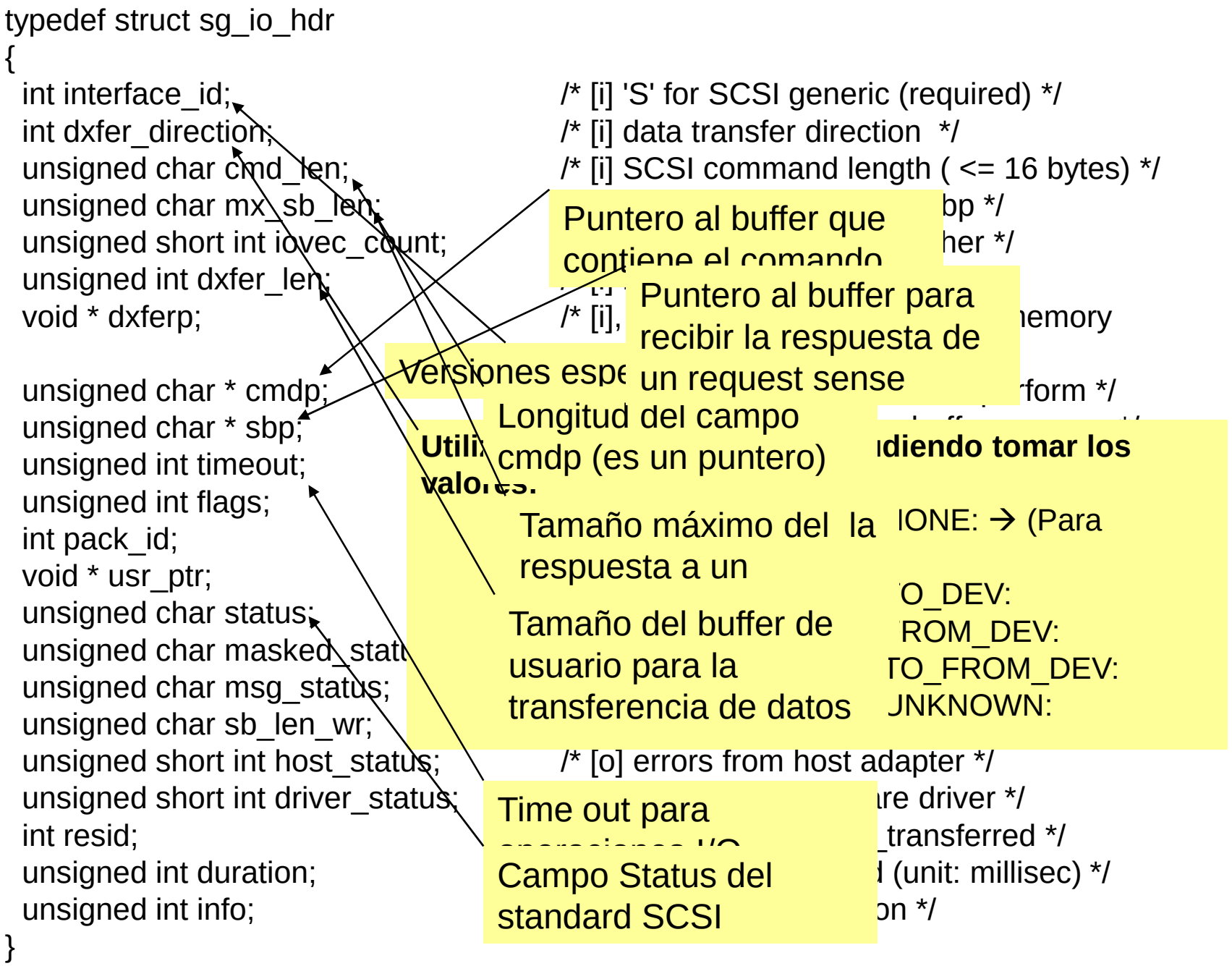
Ejemplo de envío de un “inquiry”

•Si bien una vez inicializado el dispositivo, la mayoría de los comandos enviados son Read() y Write(), nosotros enviaremos un comando “Inquiry”,

•El procedimiento es el simple manejo de drivers que conocemos de TD3

- •Abrir el dispositivo SCSI que tenemos en nuestro /dev
- •Preparar el comando completando las estructuras necesarias
- •Preparar los buffers para la respuesta del comando
- •Llamar a la función ioctl() con las estructuras y buffers iniciados
- •Cerrar el dispositivo abierto en /dev

Las estructuras mas importantes del standard SCSI se encuentran en /usr/include/scsi/sg.h


typedef struct sg_io_hdr
{
int interface_id;
int dxfer_direction;
unsigned char cmd_len;
unsigned char mx_sb_len;
unsigned short int iovec_count;
unsigned int dxfer_len;
void * dxferp;
unsigned char * cmdp;
unsigned char * sbp;
unsigned int timeout;
unsigned int flags;
int pack_id;
void * usr_ptr;
unsigned char status;
unsigned char masked_stat
unsigned char msg_status;
unsigned char sb_len_wr;
unsigned short int host_status;
unsigned short int driver_status;
int resid;
unsigned int duration;
unsigned int info;
}
/* [i] 'S' for SCSI generic (required) */
/* [i] data transfer direction */
/* [i] SCSI command length (<= 16 bytes) */
bp */
her */
/* [i],
nemory
rform */
diendo tomar los
IONE: → (Para
O_DEV:
ROM_DEV:
TO_FROM_DEV:
JNKNOWN:
/* [o] errors from host adapter */
re driver */
_transferred */
(unit: millisec) */
on */
Puntero al buffer que
Puntero al buffer para
recibir la respuesta de
un request sense
Versiones espe
Longitud del campo
cmdp (es un puntero)
Util
valor
Tamaño máximo del la
respuesta a un
Tamaño del buffer de
usuario para la
transferencia de datos
Time out para
Campo Status del
standard SCSI

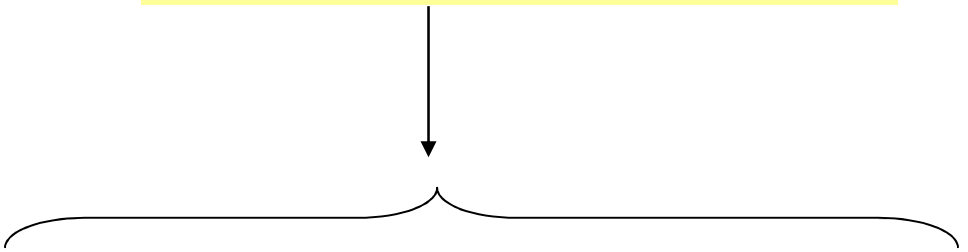
Ejemplo de código para obtener Vendor & Product

```
#include <unistd.h>
#include <fcntl.h>
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <sys/ioctl.h>
#include <scsi/sg.h>

#define INQ_REPLY_LEN 96
#define INQ_CMD_CODE 0x12
#define INQ_CMD_LEN 6

int main(int argc, char * argv[])
{
 int sg_fd, k;
 unsigned char inqCmdBlk[INQ_CMD_LEN] = {INQ_CMD_CODE, 0, 0, 0, INQ_REPLY_LEN, 0};
 unsigned char inqBuff[INQ_REPLY_LEN];
 unsigned char sense_buffer[32];
 sg_io_hdr_t io_hdr;
```

Es un standard Inquiry. por eso los campos en 0



```
if (2 != argc)
 {
  printf("Usage: 'sg_simple0 <sg_device>'\n");
  return 1;
 }
 if ((sg_fd = open(argv[1], O_RDONLY)) < 0)
  {
   perror(“dispositivo inexistente");
   return 1;
  }
 if ((ioctl(sg_fd, SG_GET_VERSION_NUM, &k) < 0) || (k < 30000))
  {
  printf("%s No es un dispositivo SCSI", argv[1]);
  return 1;
  }
  memset(&io_hdr, 0, sizeof(sg_io_hdr_t));
  io_hdr.interface_id = 'S';
  io_hdr.cmd_len = sizeof(inqCmdBlk);
  io_hdr.mx_sb_len = sizeof(sense_buffer);
  io_hdr.dxfer_direction = SG_DXFER_FROM_DEV;
  io_hdr.dxfer_len = INQ_REPLY_LEN;
  io_hdr.dxferp = inqBuff;
  io_hdr.cmdp = inqCmdBlk;
  io_hdr.sbp = sense_buffer;
  io_hdr.timeout = 20000;
```

```
if (ioctl(sg_fd, SG_IO, &io_hdr) < 0)
 {
 perror("sg_simple0: Inquiry SG_IO ioctl error");
 return 1;
 }
printf("Some of the INQUIRY command's response:\n");
printf(" %.8s %.16s %.4s\n", &inqBuff[8], &inqBuff[16], &inqBuff[32]);
printf("INQUIRY duration=%u millisecs, resid=%d\n", io_hdr.duration, io_hdr.resid);
close(sg_fd);
return 0;
}
```

Arquitecturas en infraestructuras de almacenamiento

- **DAS: Direct Attached Storage**
- **NAS Network Attached Storage**
- **SAN: Storage Area Networks**

DAS: Direct Attached Storage

- El medio de almacenamiento está directamente conectado a la unidad computacional
- Las peticiones al medio de almacenamiento son por bloques o sectores.
- Una interfase SCSI Con un array de discos podría ser un ejemplo de esto
- Una PC seria una forma simple de DAS.

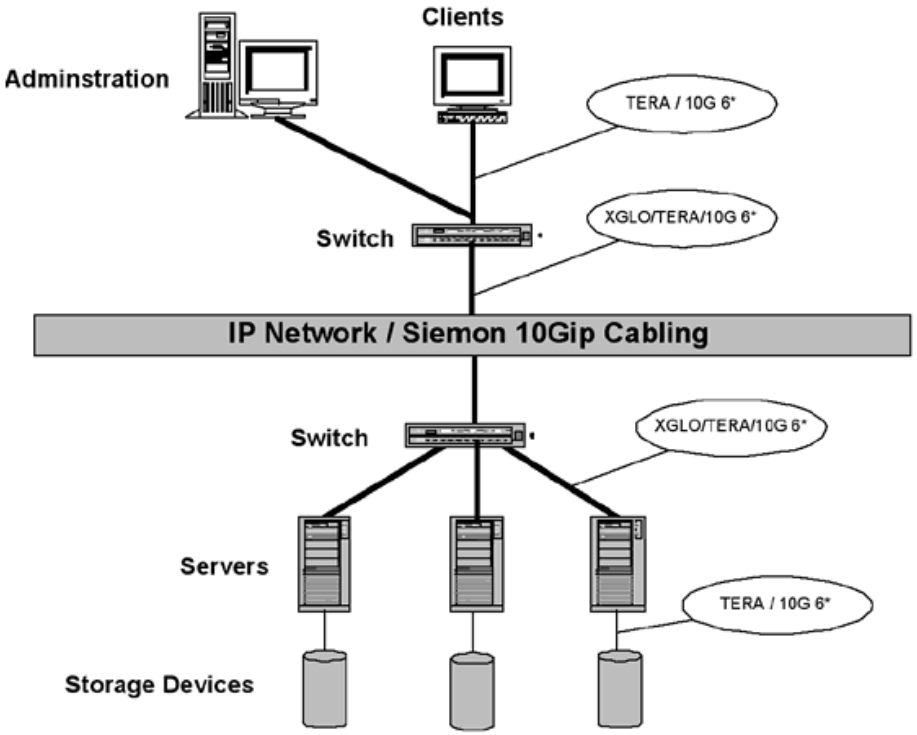


Figure 1: A simple DAS Diagram

NAS: Network Attached storage

- Acceso a nivel Archivo compartido
- Se accede al sistema de almacenamiento como un servicio de red. NFS, CIF
- Se utiliza el stack TCP/IP en la mayoria de los casos

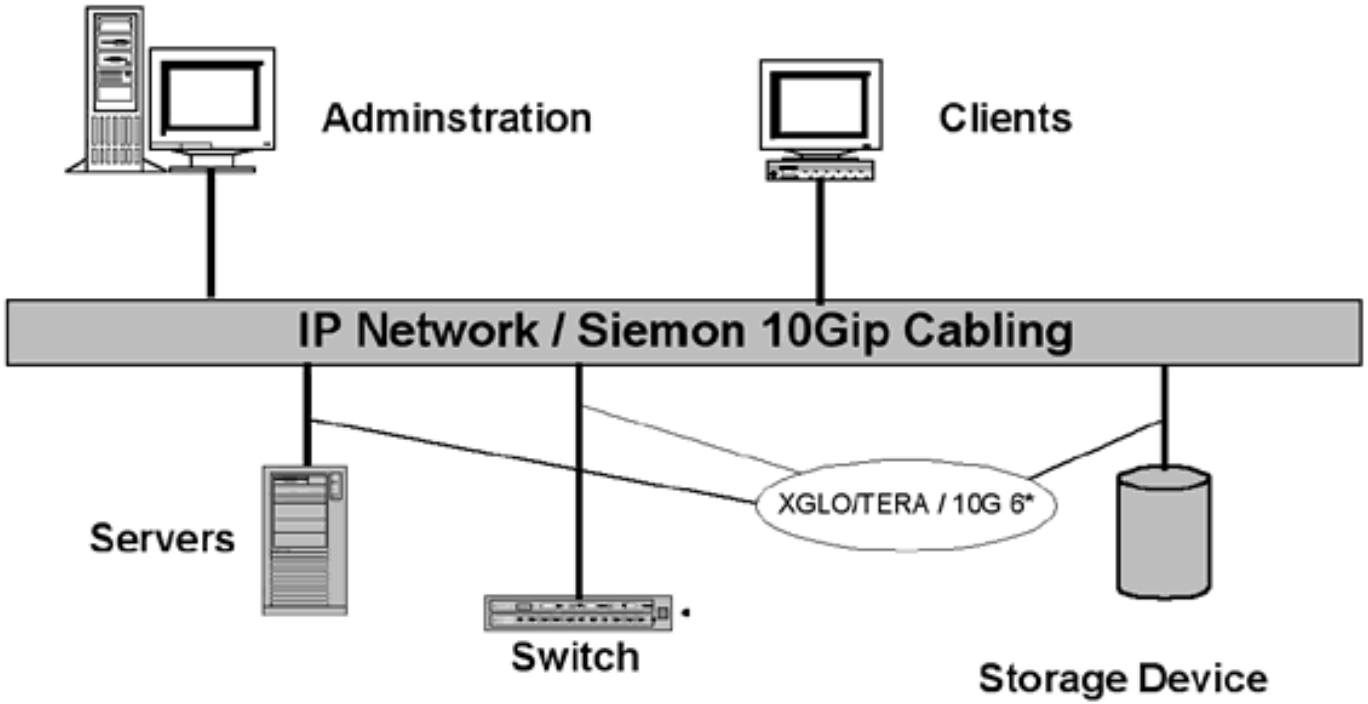


Figure 2: A simple NAS Diagram

Ejemplo NAS – NFS Filesystem

SAN: Storage Area Networks

- El almacenamiento reside en una red dedicada
- Las peticiones de entrada/salida hacen referencia a bloques o sectores de un dispositivo determinado.
- El concepto de SAN es independiente de la red que haya por debajo

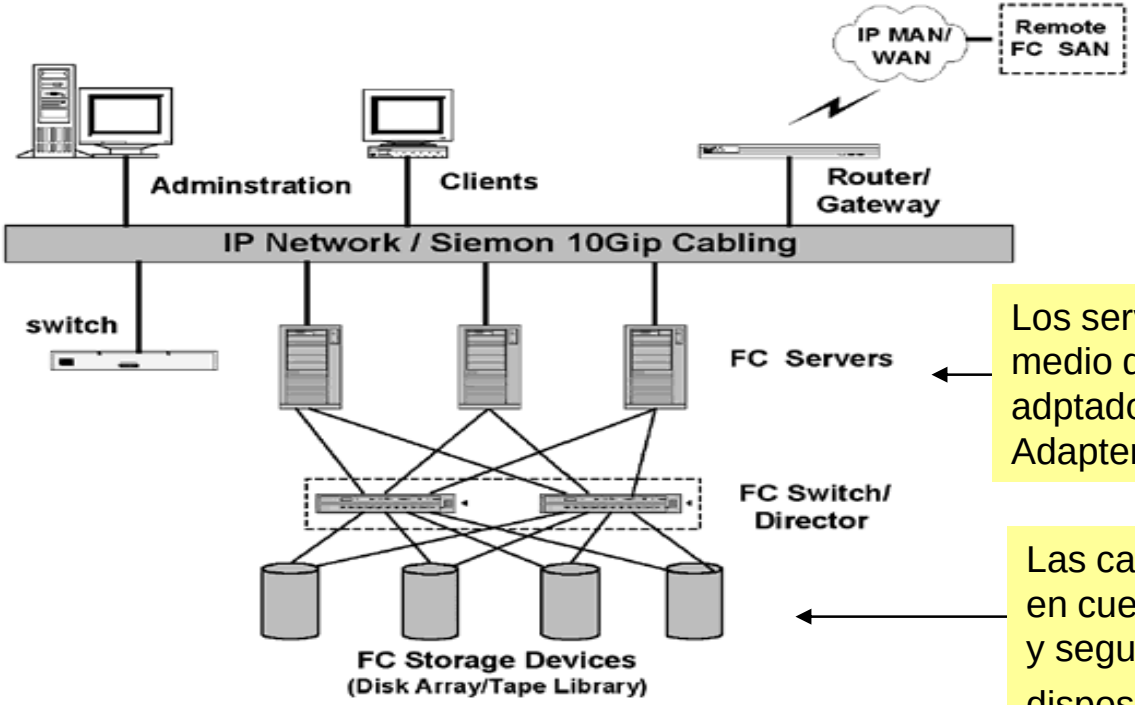
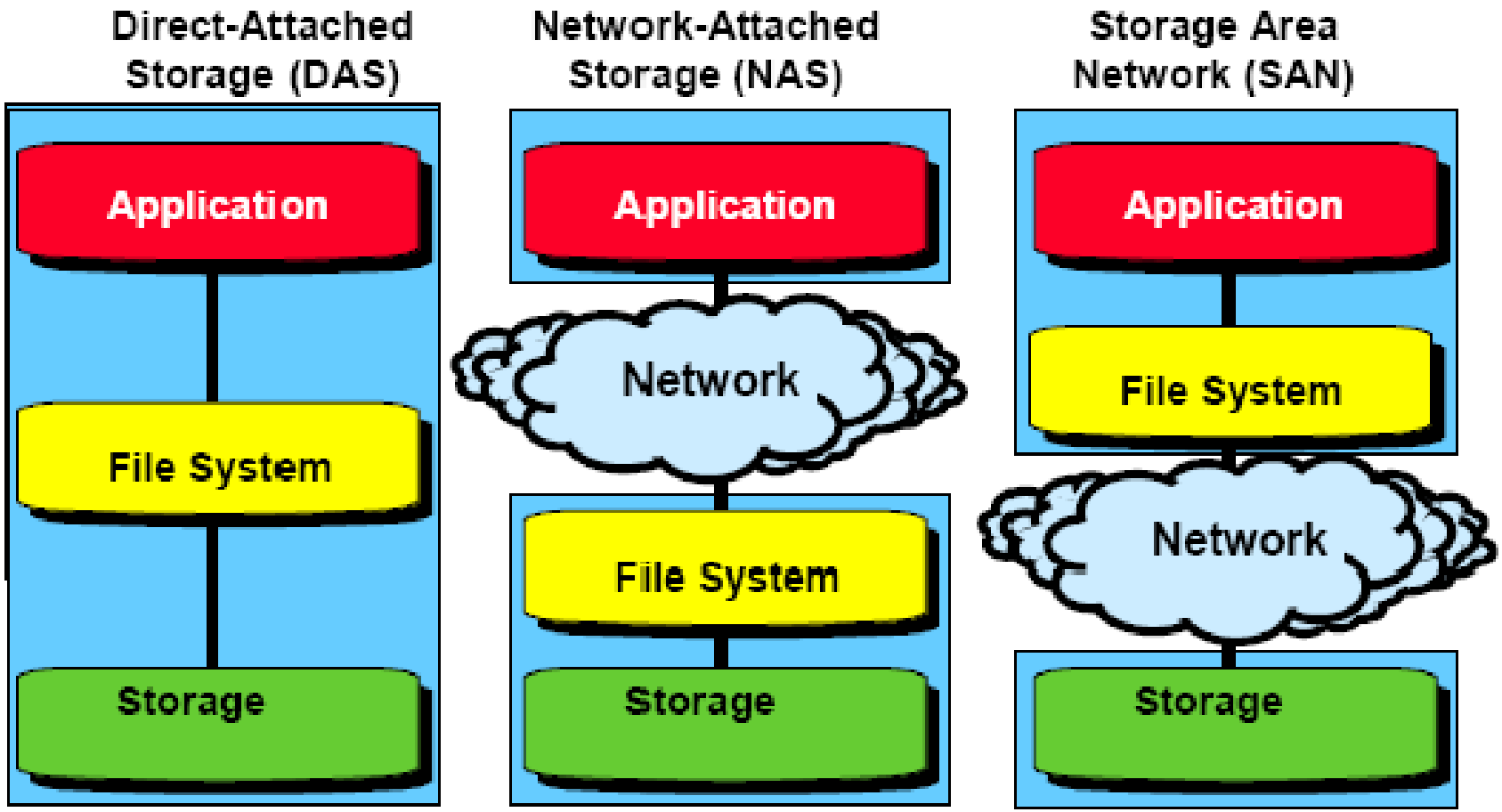


Figure 3: A Simple FCIP SAN Diagram

Los servidores se conectan a la SAN por medio de una o mas interfaces o placas adptadoras denominadas HBA (Host Bus Adapters)

Las cabinas de discos se diseñan teniendo en cuenta la importancia de la disponibilidad y seguridad de los datos contenidos en sus dispositivos

Comparación de nivel que opera en la red de almacenamiento.



Ventajas de SAN

- El sistema de almacenamiento es accesible por todos los sistemas – No existe una computadora dedicada a este fin – Reducción de costos e impacto de mantenimiento
- Facilidad para clustering – failover y programacion distribuida accediendo a los mismos datos
- Carece del overhead de NAS en LAN – No se comparten archivos sino dispositivos de bloque
- Transferencia entre dispositivos de almacenamiento

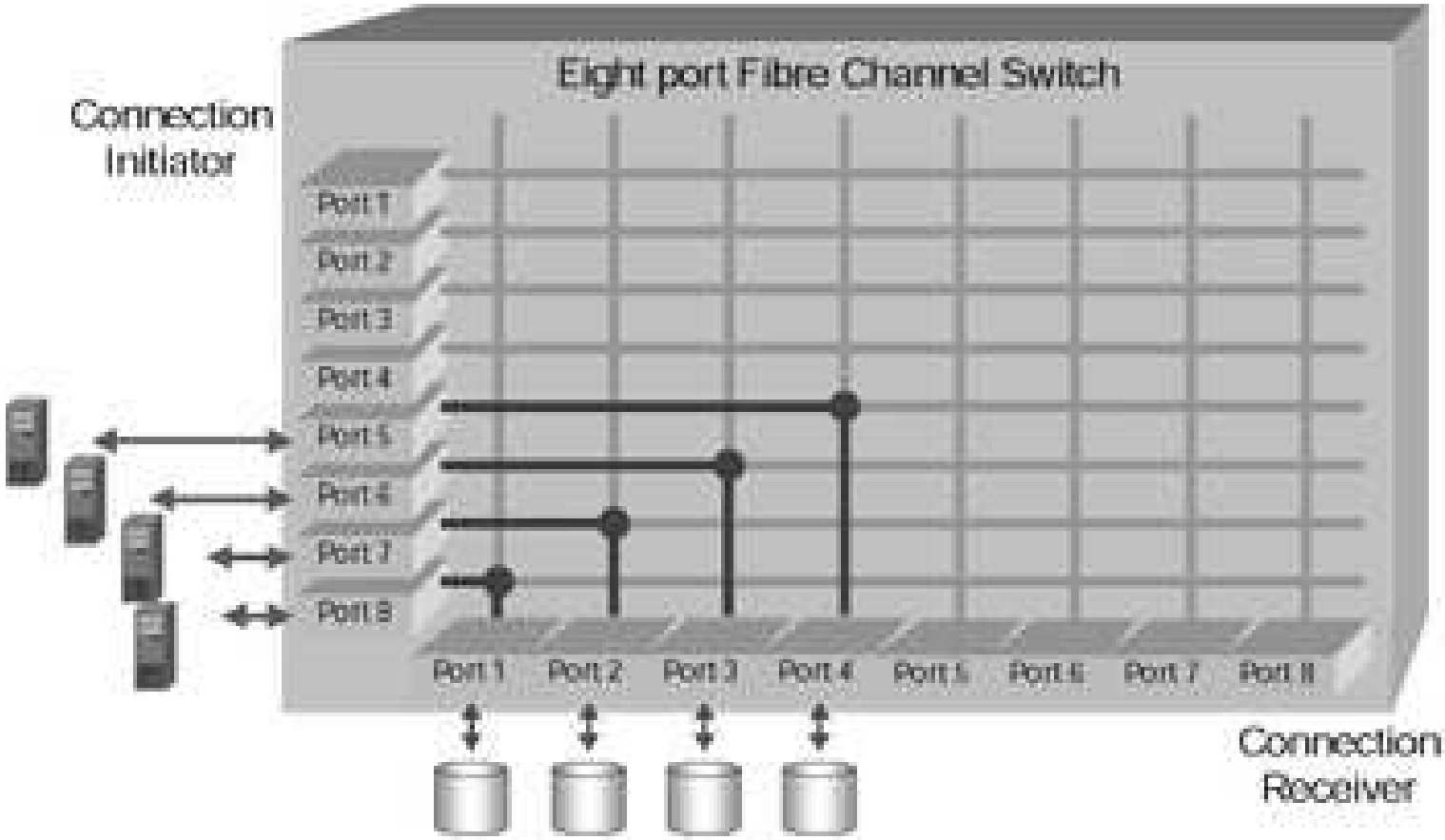
COMO HEMOS ADELANTADO EN PRESENTACIONES ANTERIORES UNA RED DIDICADA SAN, A FIN DE MAXIMIZAR SU RENDIMIENTO, UTILIZA FC COMO TRANSPORTE DEL PROTOCOLO SCSI, SIN LAS LIMITACIONES DE UNIDADES DEFINIDAS PARA SCSI NI LAS DE DISTANCIAS

Una SAN debe ser capaz de crecer junto a la necesidad de almacenamiento / procesamiento de la organización

Compartición de datos en SAN

- **A Nivel Hardware → Mediante ampliación del protocolo standard SCSI y un sistema inteligente en el subsistema o cabina de almacenamiento**
- **A Nivel Sistema operativo: Sistema de archivos distribuidos – El sistema por medio de drivers distribuye un solo File System por varios caminos diferentes. → Sitema dificil de mantener**
- **A Nivel aplicación – Base de datos**

SAN Con Switch Fabric



Diseño de la topología interna de una entidad o cabina de almacenamiento

Aspectos a evaluar

- **Capacidad**
  - Requerimiento de las aplicaciones
  - Potencial de crecimiento del sistema

Impacta en el tamaño, cant de discos y topología

- **Performance**
  - Tasa requerida por SLA
  - Tiempo de respuesta

Velocidad, Seek time, cache y topología

- **Seguridad**
  - Tolerancia a fallos
  - Tiempo de recuperación (Disaster Recovery)

MTBF y redundancia

Topologías dentro de la unidad de almacenamiento

- RAID (Redundant Array of Independent Disks)
  - Definido en 1988 en un paper de David Patterson et al, de la Universidad de California Berkeley.
  - Grupo de discos múltiples agrupados para proveer mayor rendimiento, tolerancia a fallos o ambos.
  - El objetivo principal es la recuperación inmediata de un sistema ante la falla de un disco a partir de los restantes discos del conjunto
  - Ante una situación de recuperación el sistema funcionará a modo “degradado”
  - RAID no puede mejorar el rendimiento en cuanto a tiempos de acceso
  - RAID no protege datos – Un sistema RAID → 1 Filesystem
  - Protege a nivel físico pero no puede contemplar desastres naturales
  - La topología raid no puede ser redefinida sin causar la necesidad de reconstruir la entidad lógica

- VOLUMES
  - Capa de abstracción de mayor nivel que puede incluir o no un sistema RAID
  - Permite el redimensionamiento del sistema de almacenamiento bajo demanda sin necesidad de reinicializar al mismo.

Hardware RAID

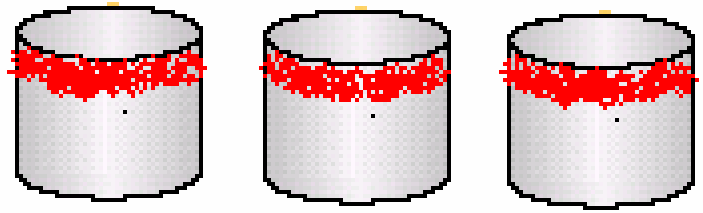
- **El sistema basado en el hardware gestiona el subsistema independientemente de la máquina y presenta a la máquina un único disco por conjunto de discos RAID.**
- **Los discos se conectan a la controladora RAID hardware.**
- **En el manejador de la controladora se define el nivel y modo de funcionamiento del RAID.**
- **Tienen lectura escritura en paralelo. Memoria caché para lectura escritura.**

Software RAID

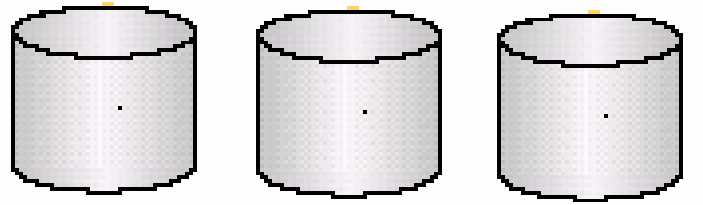
- El software RAID implementa los diversos niveles de RAID en el código del kernel (dispositivo de bloque).
- Ofrece la solución más barata ya que las tarjetas de controladores de disco o los chassis "hot-swap" suelen ser honerosos..
- El software RAID también funciona con discos IDE más baratos así como también con discos SCSI.
- Con los CPUs rápidos de hoy en día, el rendimiento del software RAID ha aumentado considerablemente con respecto al hardware RAID.

Modelo de capas RAID

Dispositivo RAID (generalmente /dev/mdx)

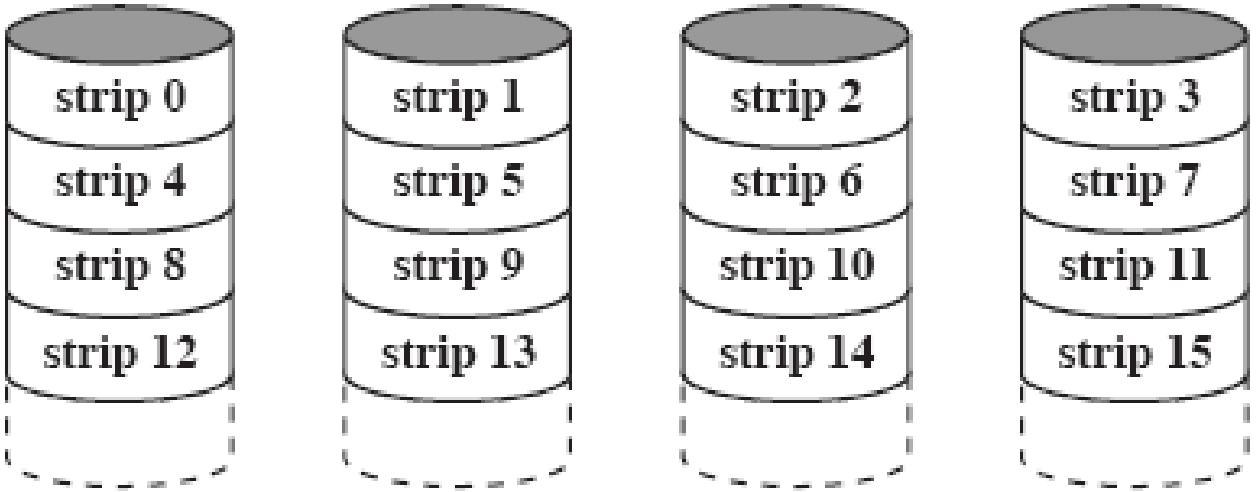


Particiones RAID (poseen metadata)



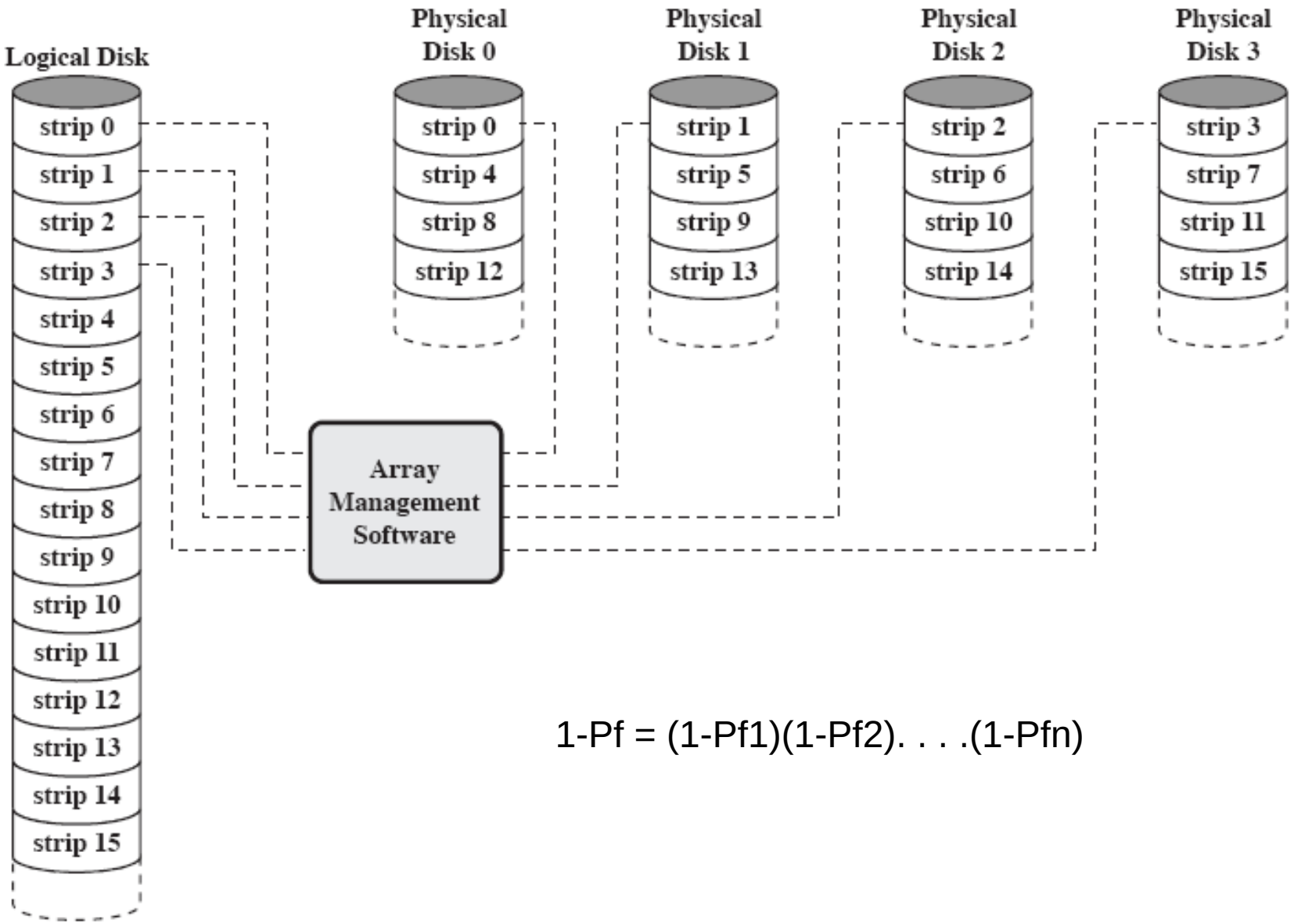
particiones físicas

RAID 0



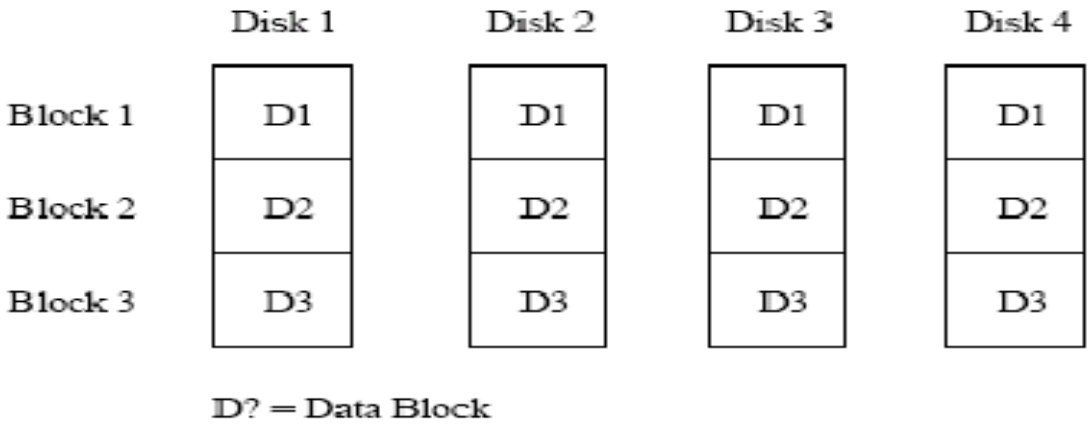
(a) RAID 0 (non-redundant)

- Se ve incrementada la capacidad dando un total siempre un tanto menor a la suma de la capacidad de los discos físicos.
- No posee tolerancia a fallos. Si falla un solo disco falla todo el RAID
- Según él hardware de control se ve incrementada la velocidad de lectura gracias al stripping (división de los datos en chunks) (No ocurre lo mismo en JBOD)
- El tiempo de acceso es el del peor escenario



$$1-Pf = (1-Pf1)(1-Pf2)\ldots(1-Pfn)$$

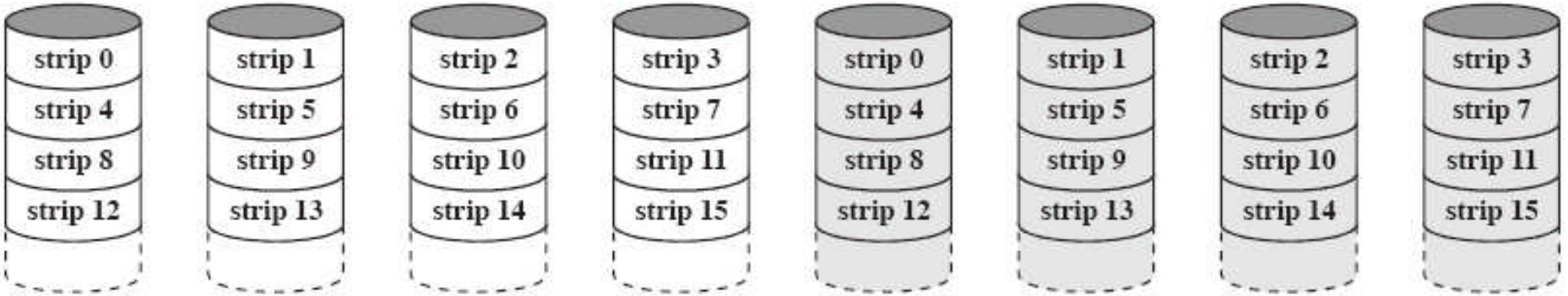
RAID 1



- Se aprovecha la capacidad de una sola copia de los datos o mejor dicho: del disco de menor capacidad
- Cada disco se espeja n veces
- Aumento de rendimiento en lectura y/o lectura concurrente
- Posibilidad de snapshot en tareas de mantenimiento
- Existe tolerancia a fallos, generación de alarmas y posibilidad de cambio en “Caliente”. (Hot Swap)

  Pf = Pf1.Pf2.Pf3……Pfn

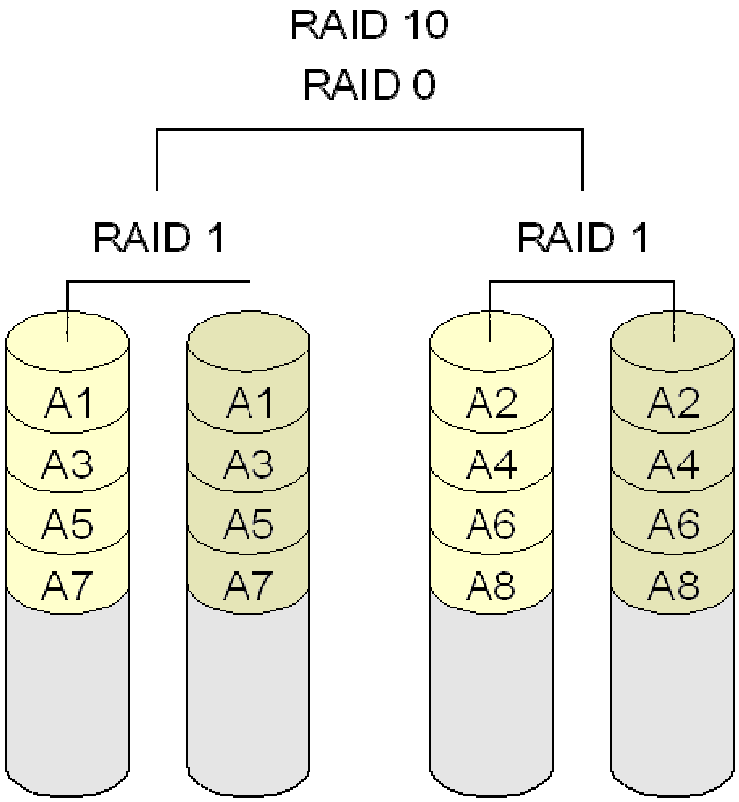
RAID 0+1



(b) RAID 1 (mirrored)

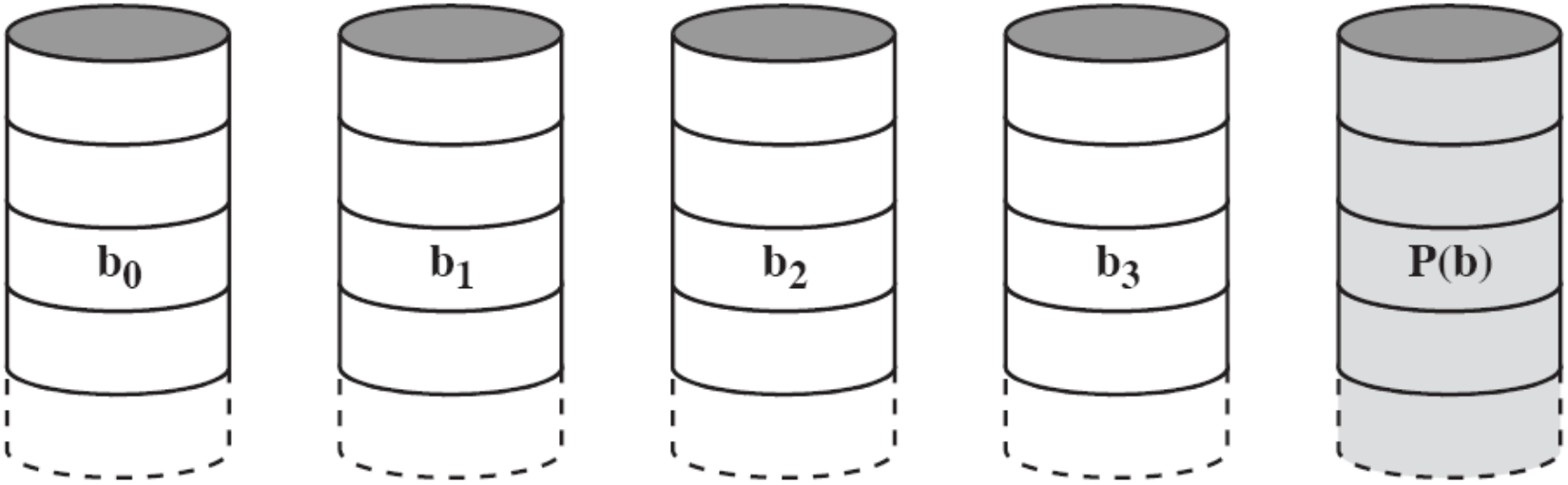
- **El nivel 0 trabaja como subsistema del nivel 1**
- **Tolera 2 fallos simultáneos pero en la misma división**
- **No se puede reconstruir un disco sin reconstruir todo la división → Reconstruccion lenta**

RAID 1+0



- El Nivel 1 trabaja como subsistema del nivel0
- Es improbable que falle el mismo disco de las diferentes divisiones primarias
- Ante la falla de un disco se reconstruye solo este a partir de dicha subdivisión Recup mas veloz que raid 0+1
- Aumento de la velocidad de escritura empleando redundancia

RAID 3 y 4



(a) RAID 3 (bit-interleaved parity)

La diferencia entre RAID 3 y RAID 4 es que el primero trabaja a nivel de bytes mientras que el segundo lo hace a nivel de bloque. Por lo que para leer un bloque no hace falta activar todos los discos

Dist disc binomial

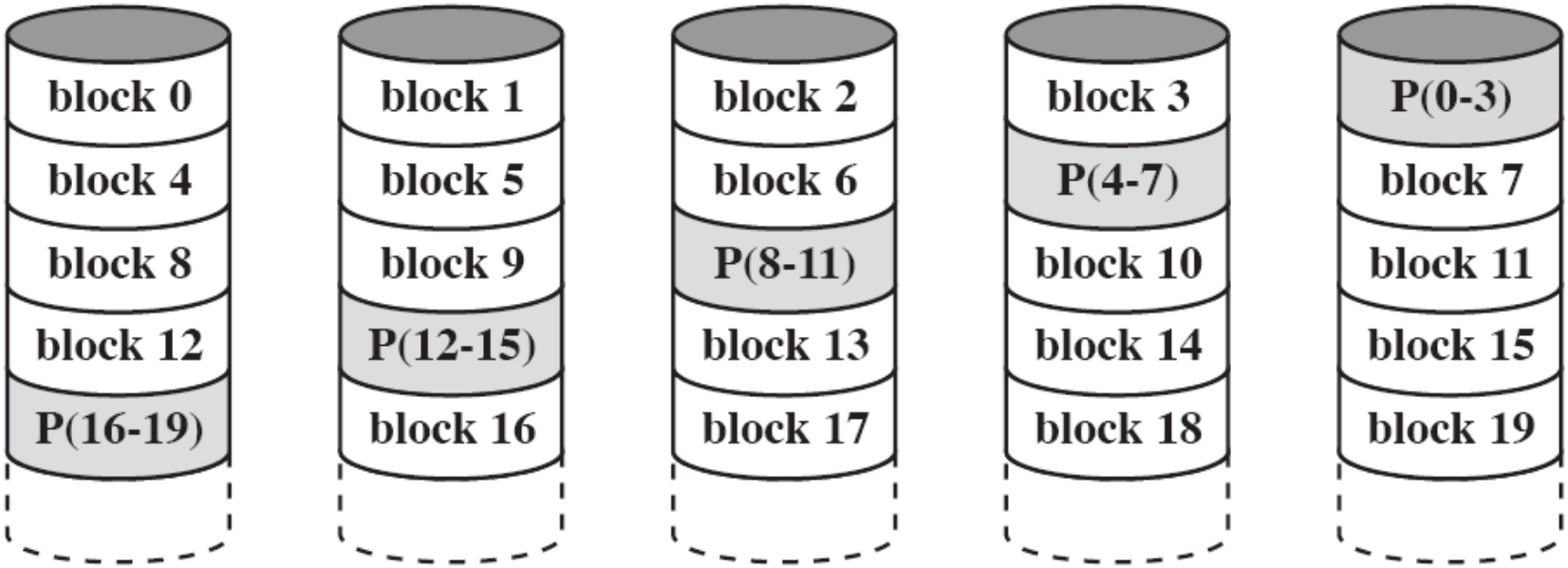
$$P(x) = \binom{n}{x} p^x (1-p)^{n-x}$$

R = (n-1)/n

$$Pf = \binom{N}{2} Pf_i^{\,2} \cdot (1-Pf_i)^{N-2} + \binom{N}{3} Pf_i^{\,3} \cdot (1-Pf_i)^{N-3} + ... + \binom{N}{N} Pf_i^{\,N} \cdot (1-Pf_i)^{N-N}$$

Logramos una Raid0 < Pf < Raid1 (Aunque aumente nunca llegamos a Pf(Raid0)

RAID 5



(c) RAID 5 (block-level distributed parity)

* Equivalente a Raid 3 en cuanto Pf

* **En caso de falla de un disc no recalcula continuamente el dato → Mas eficiente**

$$Pf = \binom{N}{2} Pf_i^{\,2} \cdot (1 - Pf_i)^{N-2} + \binom{N}{3} Pf_i^{\,3} \cdot (1 - Pf_i)^{N-3} + ... + \binom{N}{N} Pf_i^{\,N} \cdot (1 - Pf_i)^{N-N}$$

R = (n-1)/n (sin considerar discos Spare)

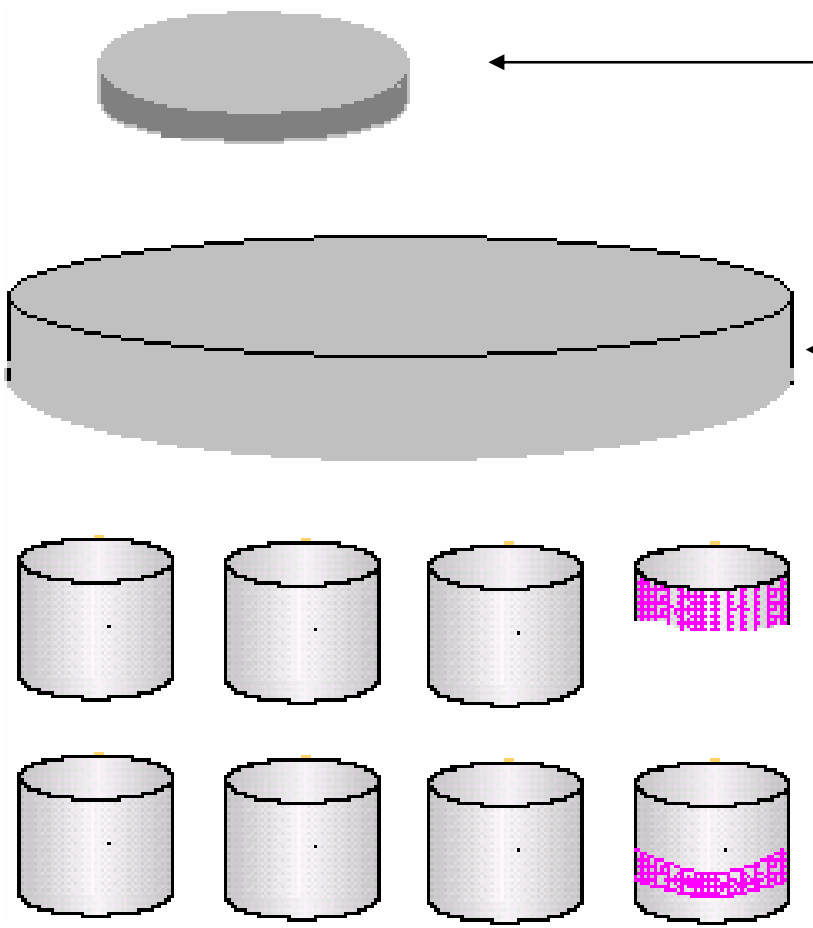
Concepto de Volume

- La gestión de volúmenes lógicos proporciona una vista de alto nivel sobre el almacenamiento en una unidad de procesamiento, en vez de la tradicional vista de discos y particiones
- Definimos como Volume a una unidad identificable para almacenamiento de datos no persistente con el dispositivo de almacenamiento. Ej. En una unidad de cintas, el cartridge es un volume.
- El concepto de volume, es totalmente independiente y por lo tanto combinable con RAID
- Una simple “caja” Linux puede manejar y administrar Volumes!!

Ventajas de Utilizar Volumes

- Los volúmenes de almacenamiento pueden ser redimensionados y movidos a voluntad
- Administrar un sistema con muchos se hace particularmente complejo si el sistema contiene discos de distintos tamaños. Balancear los requerimientos de almacenamiento puede ser una tarea muy laboriosa y compleja.
- En caso de topologías con mirroring, este puede funcionar a grandes distancias para evacuar emergencias ambientales.
- **Facilidad de Hot Relocation en el Volume Manager**
- Se pueden dedicar y etiquetar uno o varios discos físicos como Hot Spare Disks. Estos discos son administrados automáticamente (sin intervención humana) por el Volume Manager de manera que cuando un volúmen tiene una falla en uno o varios subdiscos éstos con remplazados por subdiscos nuevos creados en los Hot Spares.

Anatomía de Volumes



Logical Volume: visible como un dispositivo estándar de bloques, por lo que puede contener un sistema de archivos.

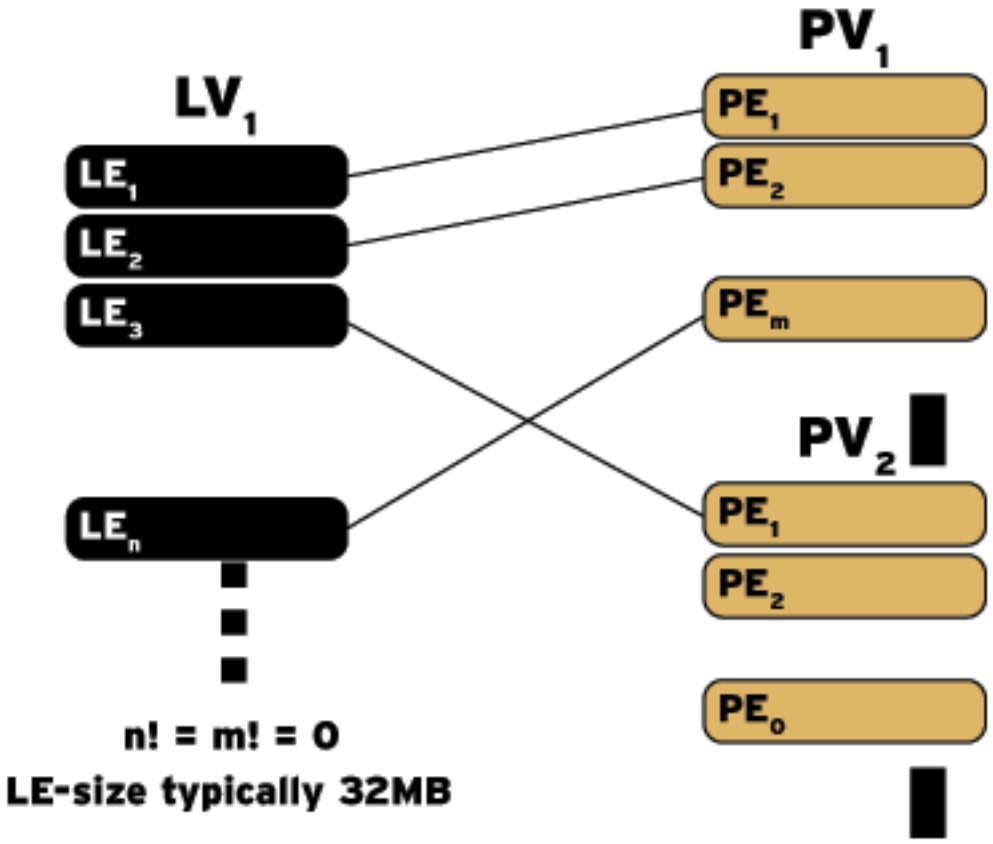
Volume groups: Se puede ver como una unidad administrativa en la que se engloban nuestros recursos

Volumes Físicos (Physical extents): unidades de bloque a nivel particiones o RAID – A este nivel se dividen los chunks de los extents generados¡

Discos o particiones físicas

El sistema lleva una asociación de physical extents utilizados para una unidad lógica con los physical extents de donde provienen. Se la conoce como “Tabla de Mapeo”

Mapeo de extents

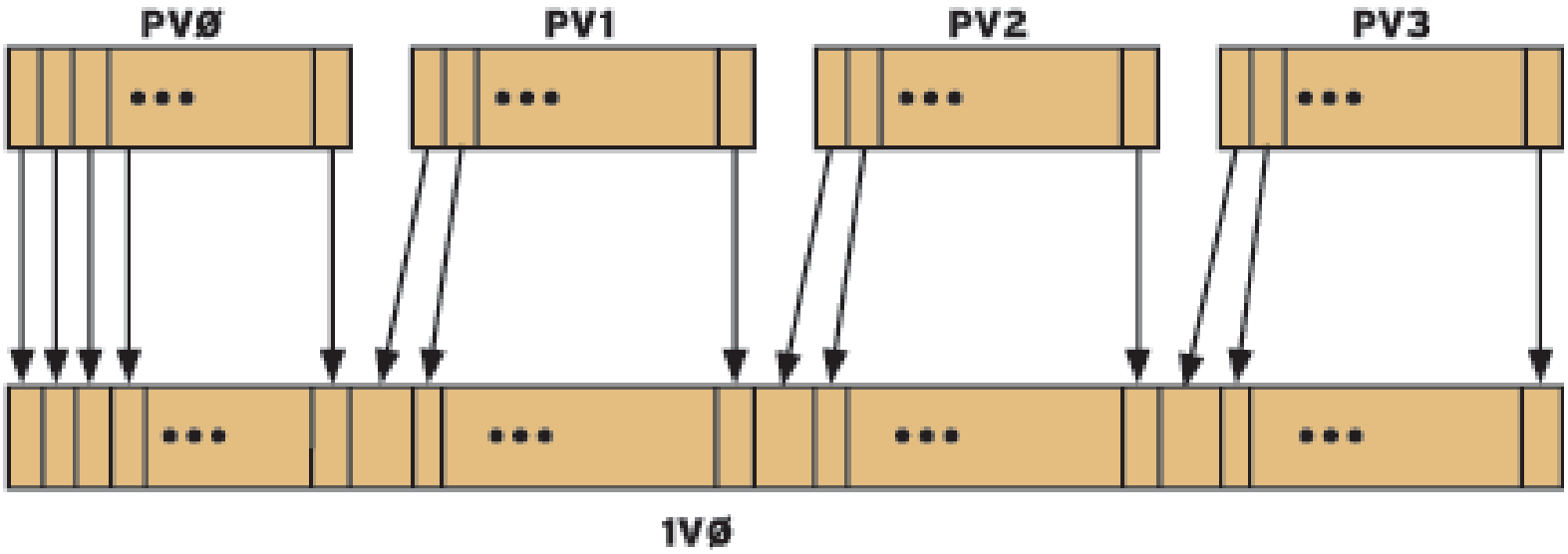


Cuando se genera un PV se divide a dicha partición física en una cantidad de physical extents de igual tamaño.

Cuando creamos un LV, lo que hacemos es asignar una determinada cantidad de pe a formar parte de una misma entidad: el LV.

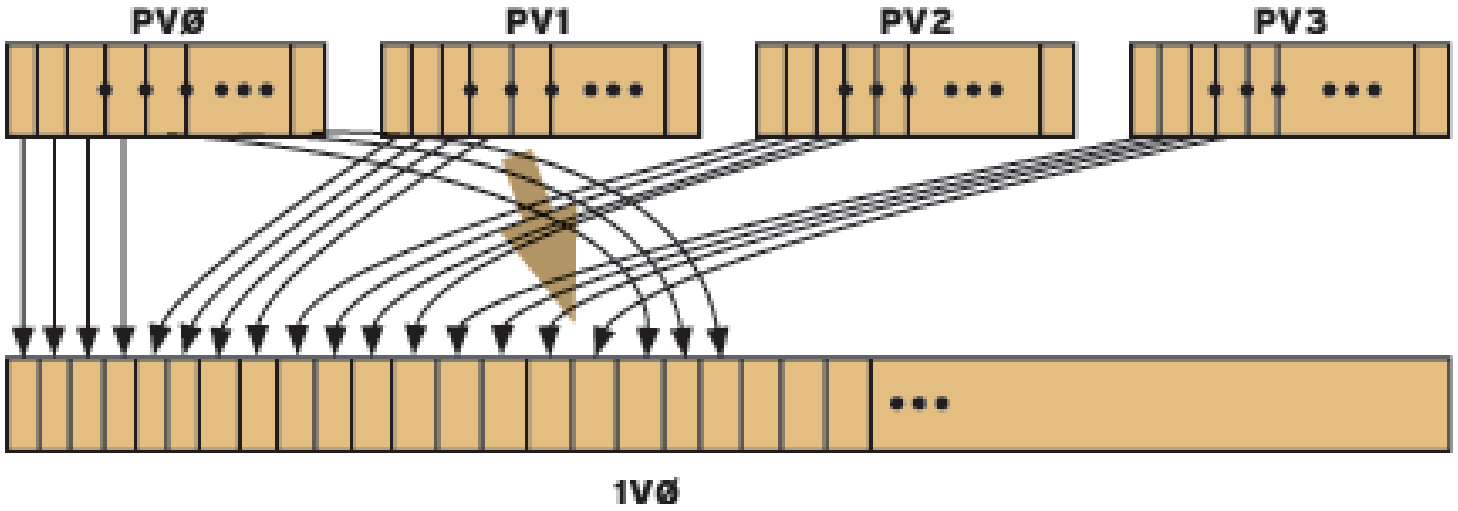
Dicha asignación se puede realizar con diferentes métricas para alcanzar diferentes prestaciones en el Filesystem

Mapeo Lineal



- Facilidad de mantenimiento para expand y shrink
- Permite generar inmensos volumenes lógicos con capacidades no existentes físicamente.
- No es optimo para lecturas paralelas simultaneas a nivel volume,

Mapeo distribuido (Striped mapping)



- Aumento del rendimiento en la lectura – Disponibilidad.
- Lento para mantenimiento expand y shrink
- Complejo para disaster recovering cuando el almacenamiento es remoto.

Modos de implementación

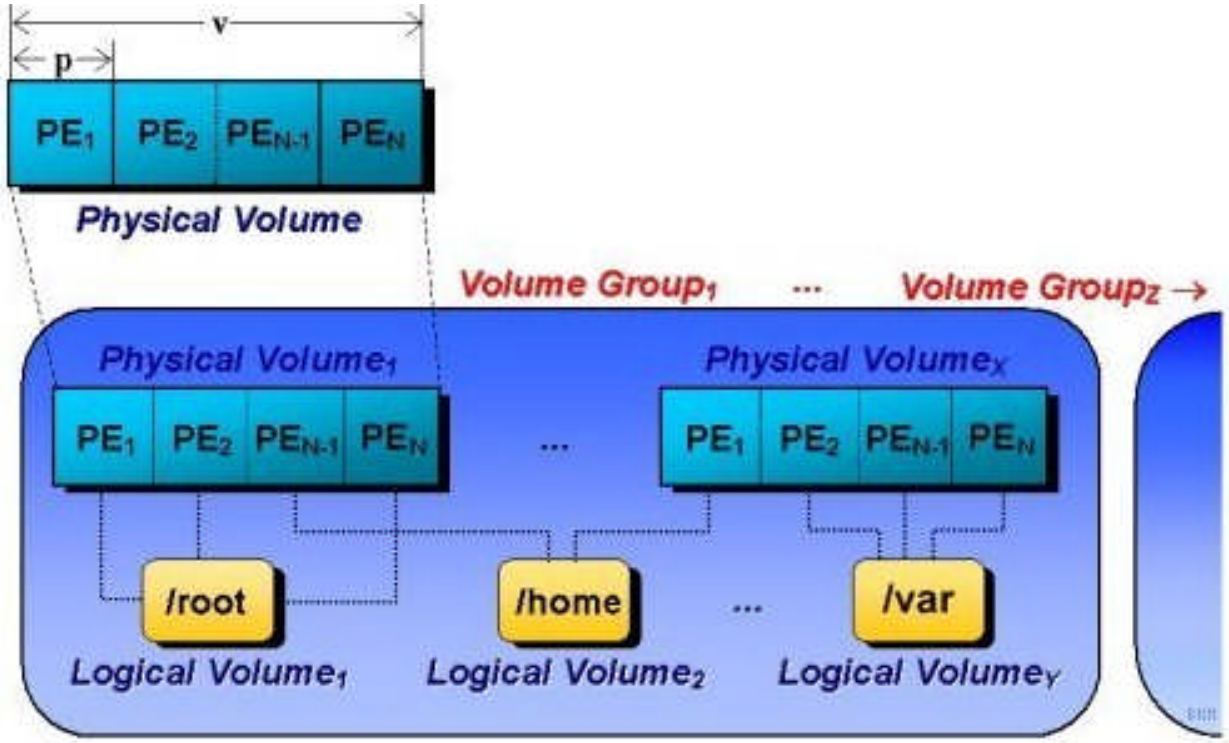
- **Volumen directo o lineal (linear volume):** sus unidades lógicas de almacenamiento están almacenadas en forma lineal sobre las unidades físicas, llenando los volúmenes físicos según sea necesario y de forma adyacente.

- **Volumen entrelazado (stripped volume):** sus unidades lógicas se distribuyen en forma alternada en distintos volúmenes físicos. Esto tiene la finalidad de aumentar el rendimiento de lectura y escritura, al hacerse al mismo tiempo sobre dispositivos físicos distintos.

- **Volumen espejado (mirrored volume):** En este tipo de volumen lógico, cada unidad lógica se escribe por duplicado en unidades físicas de distintas particiones físicas. Permite una replicación instantánea y recuperación ante errores físicos

- **Volumen de instantánea (snapshot):** Este es un volumen especial, en el que se crea una copia congelada de otro volumen, de tal forma que el otro volumen puede ser modificado, mientras que en este se puede ver cómo era el original cuando se ejecutó la instantánea. Es ideal para hacer backups en línea en sistemas que los archivos cambian, y se necesita una copia coherente.

Planificación de un sistema Unix

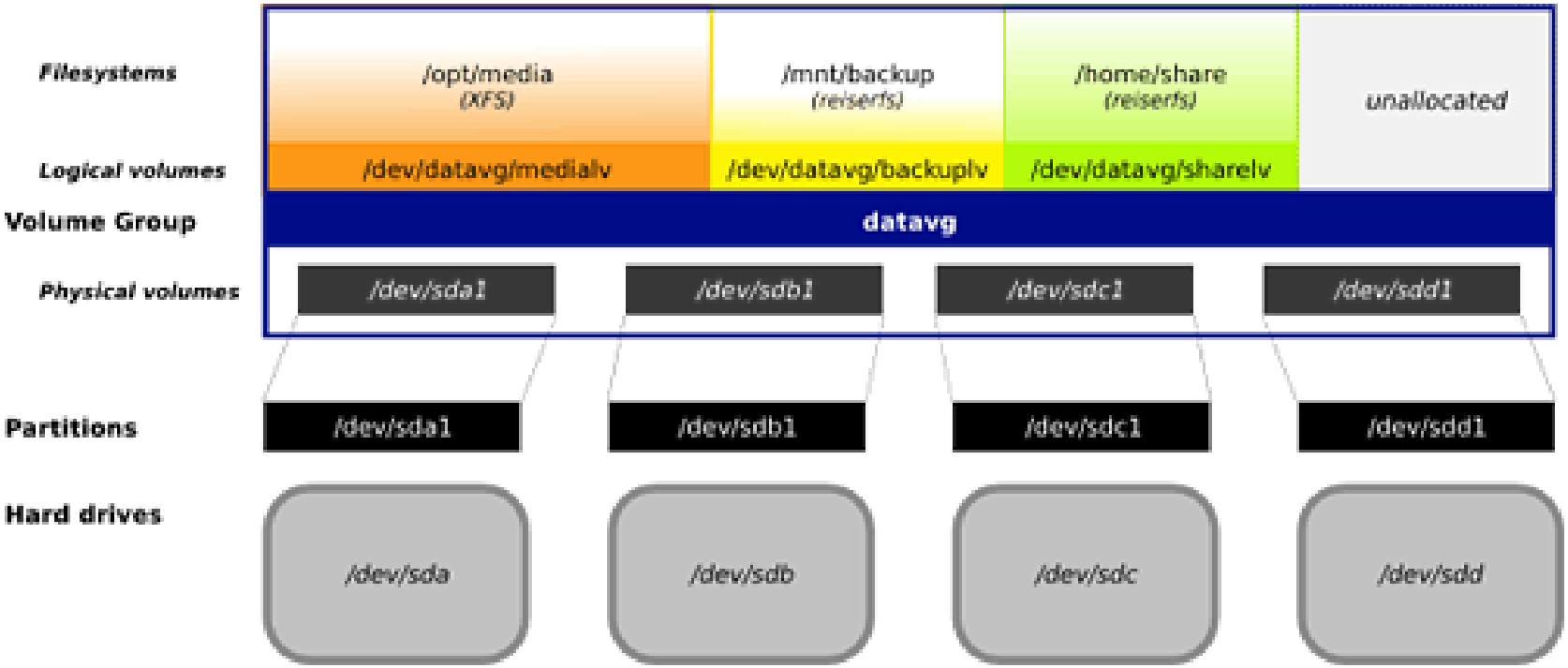


- /boot →Disco Local (Raid 1 o RAID 0)
- /etc → Disco Local
- /Home → SAN
- /Var → SAN

Espejado a nivel Volume

- El mirroring ocurre a nivel FileSystem (Se espeja un volumen lógico)+
- El Group toma la cantidad necesaria de extents para crear el volumen espejo. Es conveniente que el extent provenga de volumenes físicos diferentes.
- El espejado a nivel volume agiliza un determinado filesystem mas que otro en el mismo grupo para la lectura. (Requerimiento de aplicaciones)
- De esta manera una extension lógica queda asociada a mas extensiones físicas de las necesarias aumentando la disponibilidad de la información.

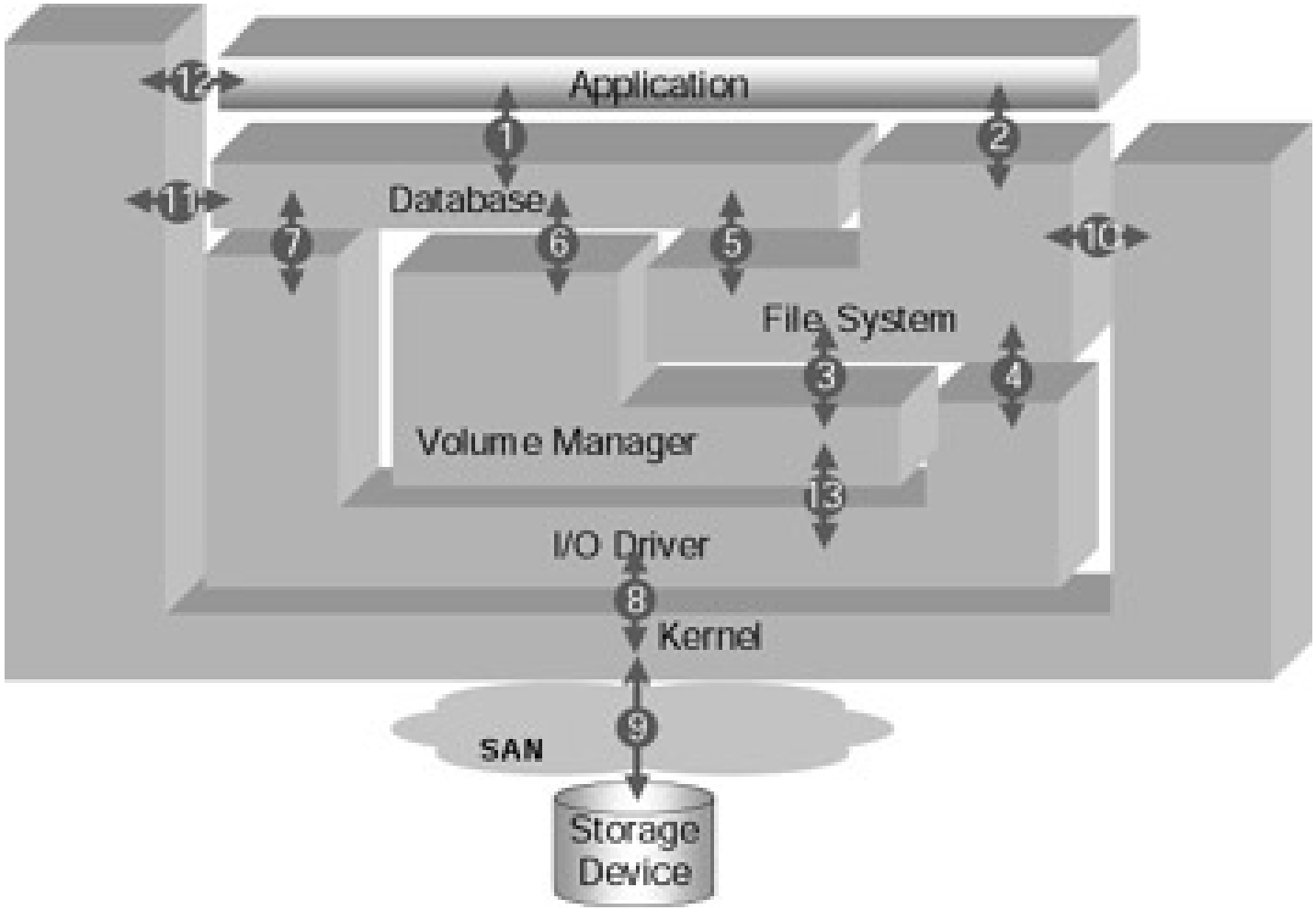
Ejemplo de aplicación



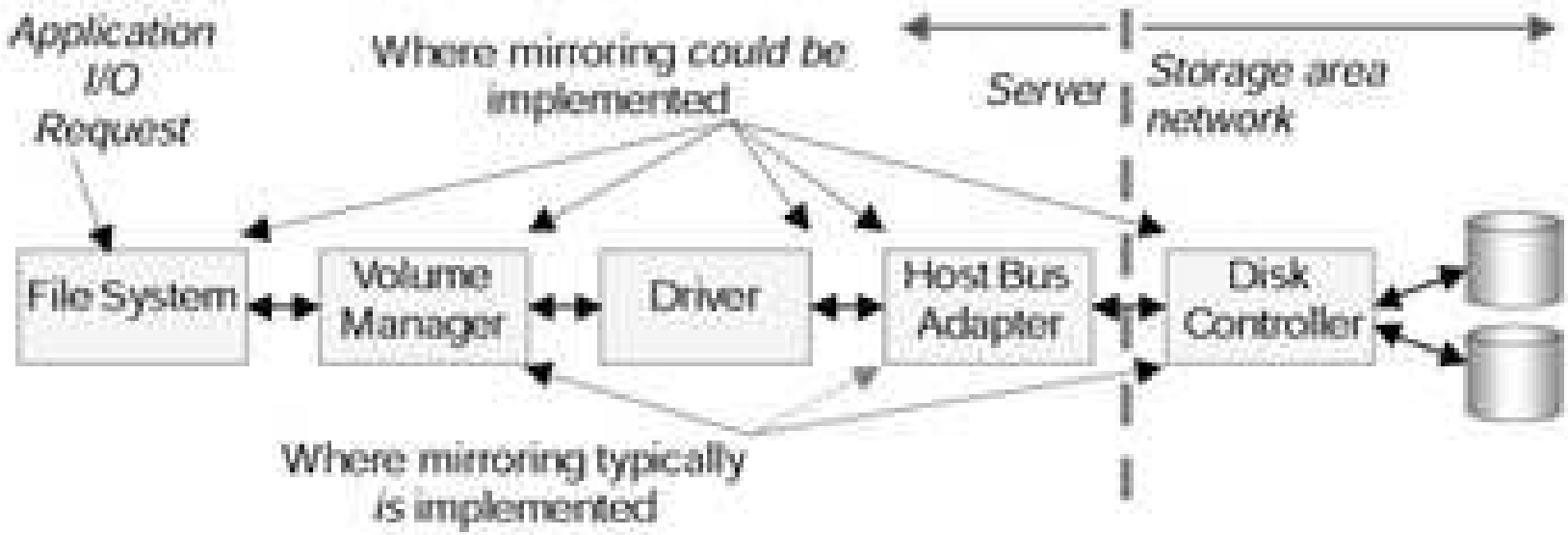
Muy importante

- Todo lo que sea “constructivo” debe realizarse desde las capas inferiores hacia las superiores.
- Todo lo que sea “destructivo” debe ser realizado desde las capas superiores hacia las inferiores

Entidades y comunicación entre las mismas



Mirroring a diferentes niveles

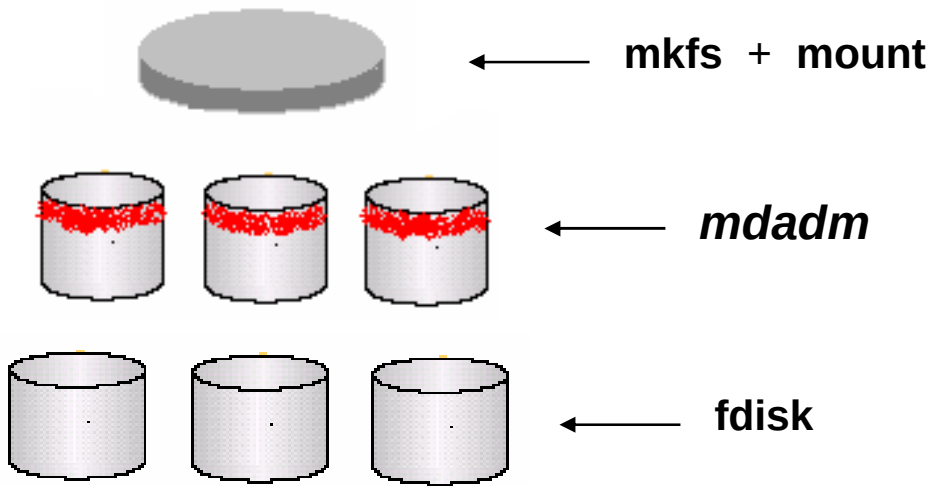


Los requerimientos de lectura de una aplicación sobre un filesystes se traducen en requerimientos al controlador de volume

El contolador de volume traduce los requerimientos a los subsistemas de discos que este contenga, que terminan siendo estos controladores de dispositivos de bloque

Software RAID en Linux

1. Creamos particiones RAID (Tipo FDh) → ***fdisk***
2. Creamos la entidad RAID en /dev indicando las particiones RAID que intervienen → ***mdadm***
3. Creamos un filesystem en el dispositivo RAID creado → ***mkfs***
4. Montamos el filesystem creado en un punto de montaje en forma temporal y/o permanente → **mount** / (archivo fstab)



LVM (Logical Volume Manager)

LVM1.0 utilizado con Kernel 2.4 & LVM 2.0 Utilizado con Kernel 2.6

- Creamos particiones LVM (Tipo 8Eh) → ***fdisk***
- Creamos los dispositivos físicos LVM → ***pvcreate***
- Creamos un VolumeGroup donde indicamos que dispositivos físicos LVM formaran parte del grupo. Por lo menos debe existir un dispositivo físico para crear un grupo. Luego se podrán agregar y quitar dispositivos dinámicamente → **vgcreate**
- Creamos uno o mas volúmenes lógicos indicando la capacidad que deseamos tomar del grupo. Si esta capacidad esta disponible el volumen lógico se creara con éxito. → **lvcreate**
- Ahora podemos crear un filesystem sobre el volumen lógico creado → **mkfs**
- Montamos el dispositivo lógico de manera que quede disponible para ser accedido.